Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **01 OCT 2014**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE
Contract-Based Integration of Cyber-Physical Analyses

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Ivan Ruchkin, Dionisio De Niz, Sagar Chaki, David Garlan

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**1**	

Contract-Based Integration of Cyber-Physical Analyses

Ivan Ruchkin, Dionisio De Niz, Sagar Chaki, David Garlan. Carnegie Mellon University, Pittsburgh, PA, USA.

Problem

CPS engineering combines diverse model-based *analyses* from various engineering domains. Differences in domain abstractions lead to integration issues:

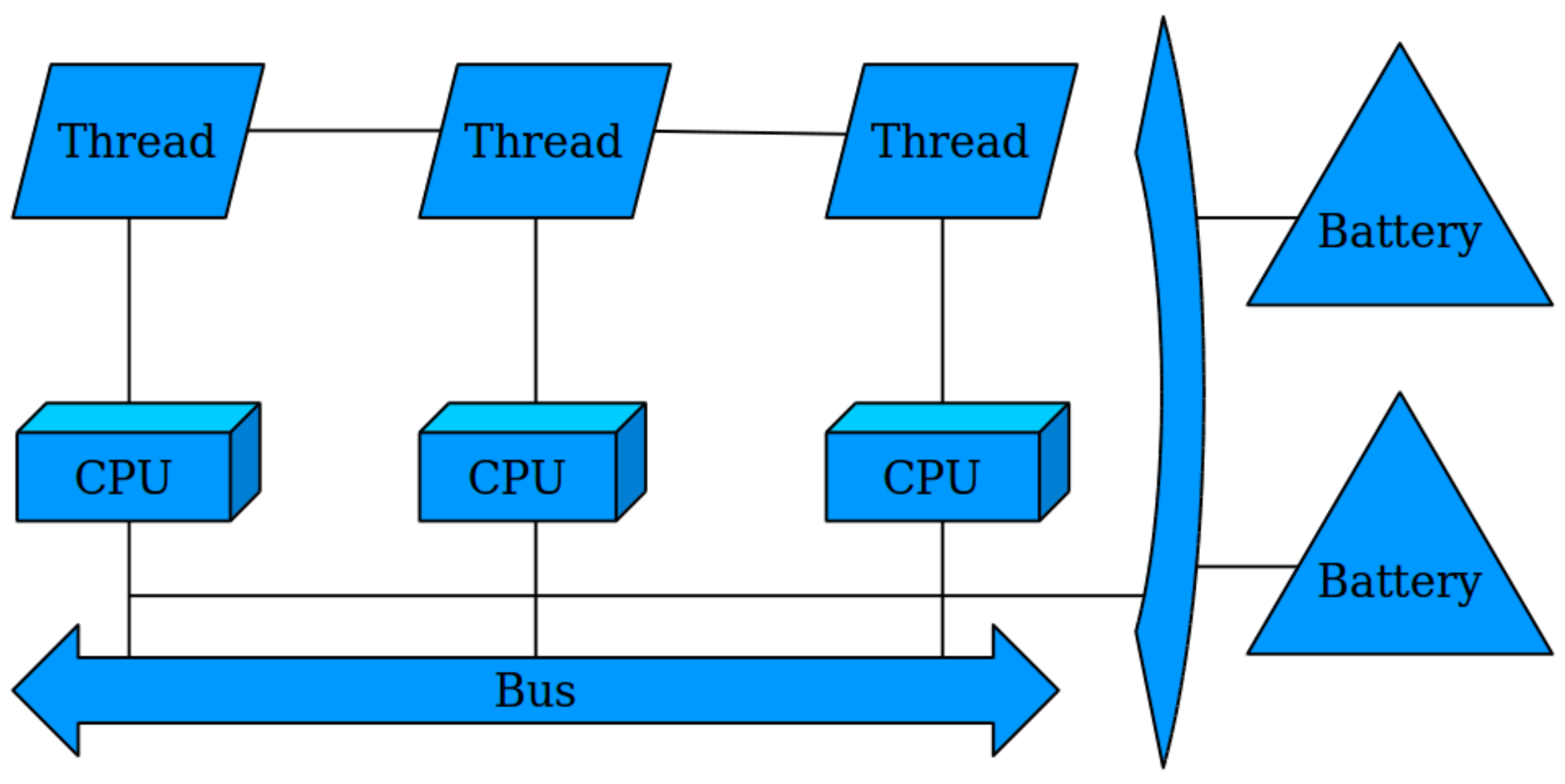
- If an assumption of an analysis are violated by another, the outputs of the former may be invalid.
- Specification of such implicit assumptions and detection of their violation is left to human designers, who are often unable to cope with complexity.
- Analysis integration problems discovered late in development lead to expensive changes to the system.

Hence the research question:

- How to specify analysis compositions and verify their correctness?

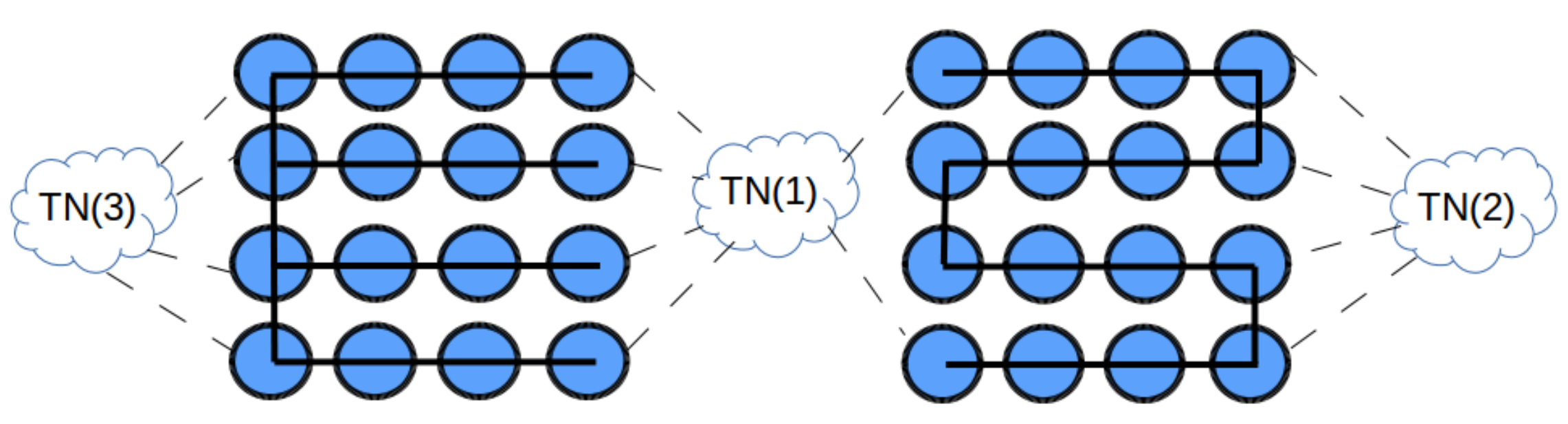
Example System

Consider an autonomous aircraft as an example CPS. It operates data with different classes of security, from normal to top secret (ThSecCl). Periodic threads (*T*) execute on several processors (*C*). The aircraft is powered with multi-cell reconfigurable batteries (*B*). The system's architecture shown below is specified in AADL.



A battery has a matrix of cells, and each cell has a current level of charge. A battery scheduler determines parallel and sequential connections between groups of cells in order to satisfy voltage and current output requirements.

Thermally, each cell exchanges heat with its neighboring cells (*thermal neighbors,* TN) through an electrical connector, affecting the risk of a *thermal runaway*.



Verification Domains

A verification domain $\sigma = (\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, [\![]\!]_\sigma)$ formalizes domain-specific constructs for several related analyses.

- $\mathcal{A}$ — a set of sorts, comprised of system elements and standard sorts. E.g., integers $\mathbb{Z}$, threads T, or scheduling policies SchedPol.
- $\mathcal{S}$ — a set of static functions that encode design-time properties. E.g., thread period Per, thread-to-CPU binding CPUBind, and system-wide Voltage.
- $\mathcal{R}$ — a set of runtime functions that encode dynamic properties. E.g., preemption relation canPrmpt(t_1, t_2) and number of cells in a battery b with i thermal neighbors TN(b, i).
- $\mathcal{T}$ — execution semantics of σ — a set of sequences of assignments to $\mathcal{R}$. We use Promela programs to implement the semantics.
- $[\![]\!]_\sigma$ — a domain interpretation of $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{T}$. E.g., $[\![\mathrm{SchedPol}]\!]_\sigma$ = {RMS, DMS, EDF}.

Formally, an AADL architectural model **m** is an interpretation $[\![]\!]_m$ of $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{T}$. E.g., $[\![T]\!]_m$ = { SensorSample, Ctrl_1, Ctrl_2 }, $[\![\mathbf{CPUBind}]\!]_m$ = { (Ctrl_1, CPU_1), (Ctrl_2, CPU_2), …) }.

$[\![]\!]_\sigma \cup [\![]\!]_{\mathbf{m}}$ form a full interpretation of $\mathcal{A}$, $\mathcal{S}$, $\mathcal{R}$, and $\mathcal{T}$.

Analysis Contracts

Each analysis is assigned a *contract* — a tuple (*I, O, A, G*).

- Inputs $I \subseteq \mathcal{A} \cup \mathcal{S}$ declare elements that the analysis reads.
- Outputs $O \subseteq \mathcal{A} \cup \mathcal{S}$ declare elements that the analysis writes.
- Assumptions $A \subseteq \mathcal{F}_\sigma$ are logical statements that must be satisfied by every input model to the analysis: $\mathbf{m} \vDash A$.
- Guarantees $G \subseteq \mathcal{F}_\sigma$ are logical statements that must be satisfied by every output model of the analysis: $\mathbf{m} \vDash G$.

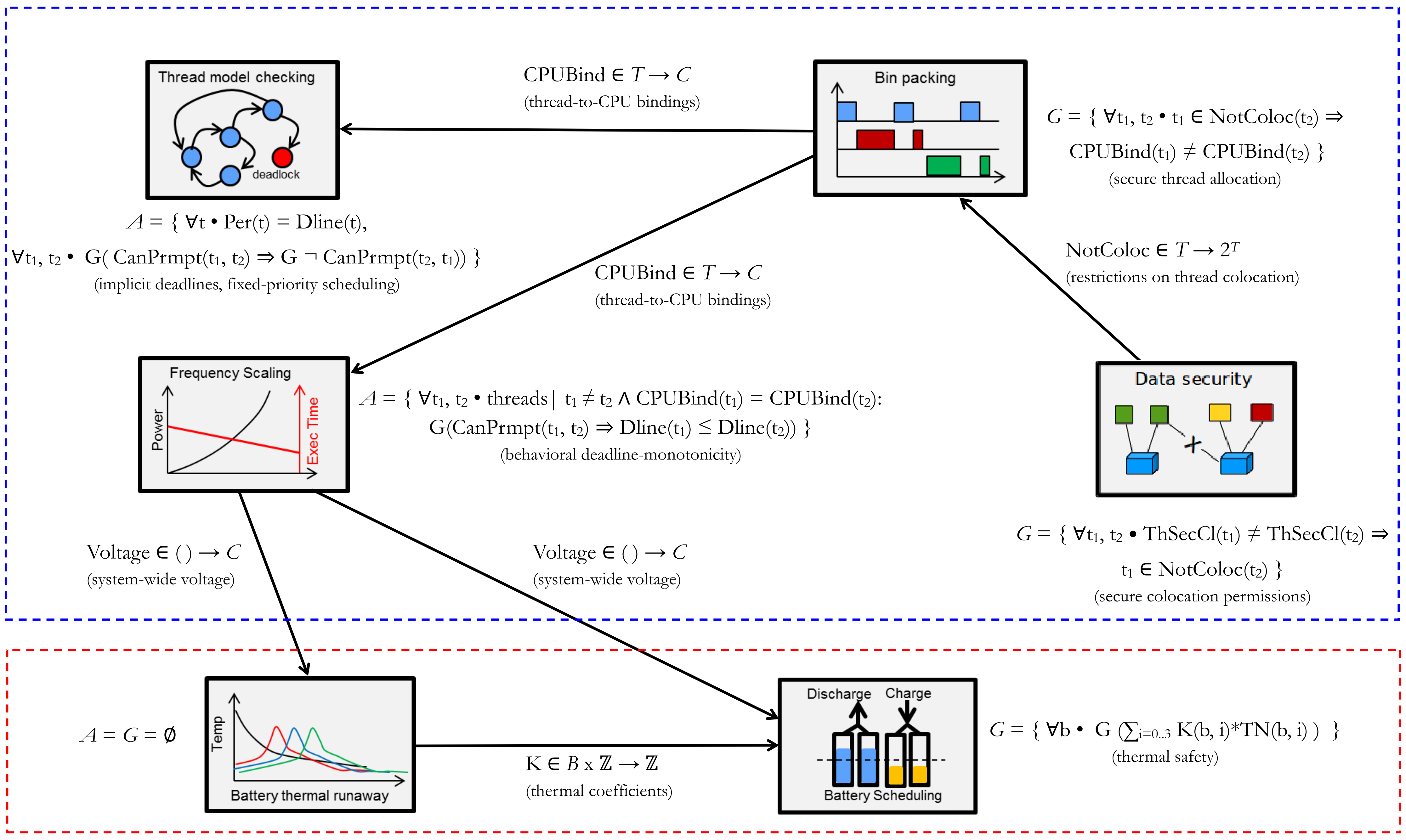
Assumption and guarantee formulas have the following syntax:

$$\mathcal{F}_\sigma ::= \forall\, v_1 \ldots v_j \bullet \varphi \mid \exists\, v_1 \ldots v_j \bullet \varphi \mid$$
$$\forall\, v_1 \ldots v_j \bullet \varphi : \psi \mid \exists\, v_1 \ldots v_j \bullet \varphi : \psi,$$

where φ is a predicate logic formula over $\mathcal{A} \cup \mathcal{S}$, ψ is an LTL formula over $\mathcal{A} \cup \mathcal{S} \cup \mathcal{R}$.

Example Analyses

Scheduling verification domain σ_{sched}



CPUBind ∈ $T \rightarrow C$
(thread-to-CPU bindings)

G = { ∀t_1, t_2 • t_1 ∈ NotColoc(t_2) ⇒ CPUBind(t_1) ≠ CPUBind(t_2) }
(secure thread allocation)

A = { ∀t • Per(t) = Dline(t), ∀t_1, t_2 • G(CanPrmpt(t_1, t_2) ⇒ G ¬ CanPrmpt(t_2, t_1)) }
(implicit deadlines, fixed-priority scheduling)

CPUBind ∈ $T \rightarrow C$
(thread-to-CPU bindings)

NotColoc ∈ $T \rightarrow 2^T$
(restrictions on thread colocation)

A = { ∀t_1, t_2 • threads | $t_1 \neq t_2$ ∧ CPUBind(t_1) = CPUBind(t_2): G(CanPrmpt(t_1, t_2) ⇒ Dline(t_1) ≤ Dline(t_2)) }
(behavioral deadline-monotonicity)

Voltage ∈ ( ) → C
(system-wide voltage)

Voltage ∈ ( ) → C
(system-wide voltage)

G = { ∀t_1, t_2 • ThSecCl(t_1) ≠ ThSecCl(t_2) ⇒ t_1 ∈ NotColoc(t_2) }
(secure colocation permissions)

$A = G = \emptyset$

K ∈ B x $\mathbb{Z} \rightarrow \mathbb{Z}$
(thermal coefficients)

G = { ∀b • G ($\sum_{i=0..3}$ K(b, i)*TN(b, i)) }
(thermal safety)

Battery verification domain σ_{batt}

Analysis Ordering

Correct execution of analyses requires satisfaction of all input-output dependencies for each analysis. Formally, contract Ci depends on contract C_j if $C_i.I \cap C_j.O \neq \emptyset$.

Am ordering $<C_1 \ldots C_n>$ of contracts is sound if and only if predecessors are not dependent on successors:

$$\forall i \in [1, n] \bullet \forall j \in [1, i) \bullet C_j.I \cap C_i.O = \emptyset.$$

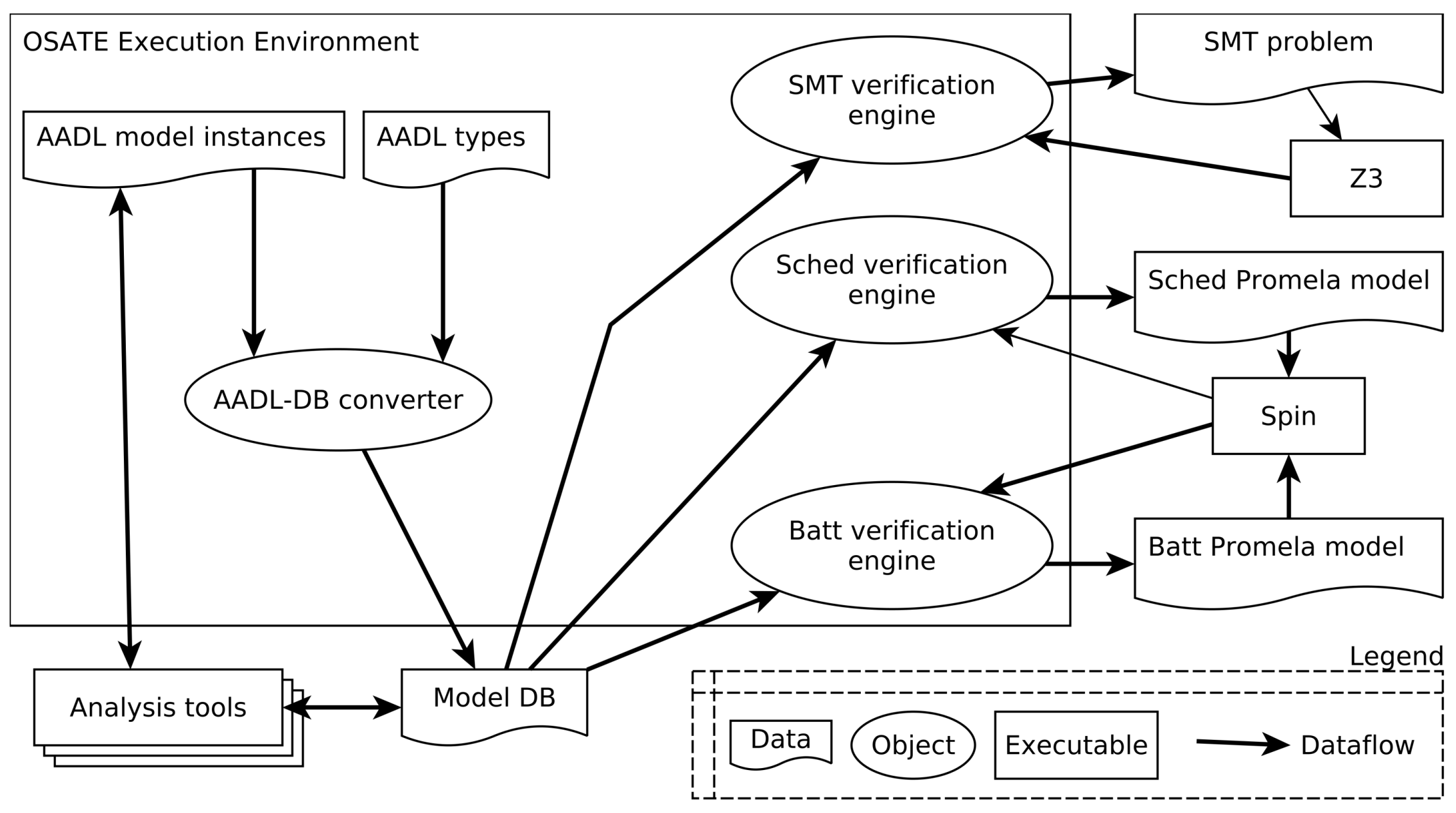
Consider a graph with vertices being contracts and edges being contract dependencies. There exists a sound ordering of contracts if and only if the graph is not cyclic. If it is not cyclic, any topological ordering is sound.

Contract Verification

The goal of contract verification is to decide $\mathbf{m} \vDash \mathcal{F}_\sigma$.

For purely first-order formulas that contain only φ, we decide satisfiability via SMT solving. An SMT program is generated based on $\mathcal{A}$ and $\mathcal{S}$ mentioned in φ, and an SMT solver is invoked on ¬ φ (or φ for existential quantification). A universally (existentially) quantified contract is satisfied if and only if UNSAT (SAT) is returned.

For formulas combining predicate formula φ and LTL formula ψ, we first generate an SMT program for φ and find all valuations of $v_1 \ldots v_j$ that satisfy φ. For each such valuation we call Spin on a Promela program that implements $\mathcal{T}$ for **m** in the domain of ψ. Formula ψ is transformed into an LTL property specification in Promela. A universally (existentially) quantified contract is satisfied if and only if the LTL property holds for all (at least one) valuations. The architecture of our verification tool is shown below:



Experimental Results

Effectiveness: we have been able to detect analysis integration errors and verify their absence for each analysis in the example.

Scalability: the results of scalability experiments with our implementations of $\mathcal{T}$ are shown in the tables below.

$\mathcal{T}_{sched}$:

Threads	(R/D)MS time*	EDF time*
3	0.01	0.01
4	0.01	0.52
5	0.07	33.4
6	0.37	2290.0
7	2.18	memlim
8	12.4	memlim
9	71.2	memlim
10	421	memlim
11	memlim	memlim

$\mathcal{T}_{batt}$:

Cells	FGURR time*	FGWRR time*	GPWRR time*
9	0.13	0.15	0.15
12	0.61	2.34	3.94
16	44	31.4	127
20	1060	619	memlim
25	memlim	memlim	memlim

* All times are in seconds.

Carnegie Mellon University

isr institute for SOFTWARE RESEARCH

Software Engineering Institute